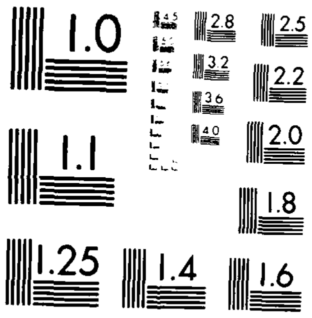

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

ALGORITHMS FOR COMPUTING THE LAG FUNCTION

G. H. Peebles and S. J. Giner

Technical Memorandum
File No. TM 81-82
27 March 1981
Contract N00024-79-C-6043

Copy No. 8

The Pennsylvania State University
Intercollege Research Programs and Facilities
APPLIED RESEARCH LABORATORY
Post Office Box 30
State College, Pa. 16804

NAVY DEPARTMENT

NAVAL SEA SYSTEMS COMMAND

ALGORITHMS FOR COMPUTING THE LAG FUNCTION

G. H. Peebles and S. J. Giner

Technical Memorandum
File No. TM 81-82
27 March 1981
Contract N00024-79-C-6043

Copy No. 8

The Pennsylvania State University
Intercollege Research Programs and Facilities
APPLIED RESEARCH LABORATORY
Post Office Box 30
State College, PA 16804

NAVY DEPARTMENT

NAVAL SEA SYSTEMS COMMAND

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TM 81-82	2. GOVT ACCESSION NO. AD-A144 862	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ALGORITHMS FOR COMPUTING THE LAG FUNCTION		5. TYPE OF REPORT & PERIOD COVERED Technical Memorandum
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) G. H. Peebles and S. J. Giner		8. CONTRACT OR GRANT NUMBER(s) N00024-79-C-6043
9. PERFORMING ORGANIZATION NAME AND ADDRESS Applied Research Laboratory/Penn State Univ. Post Office Box 30 State College, PA 16804		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Sea Systems Command, Code NSEA 63R31 Department of the Navy Washington, DC 20362		12. REPORT DATE 27 March 1981
		13. NUMBER OF PAGES 46
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release. Distribution unlimited.
Per NAVSEA - 7 August 1984.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Lag Function
Algorithm
Numerical Integration

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This memorandum provides a scheme for the numerical solution of the "Lag Function," a complex function of a parameter k. The solution involves the calculation of two double integrals of infinite extent, highly oscillatory, and with singularities at the end points.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

1/

From: G. H. Peebles and S. J. Giner

Subject: Algorithms for Computing the Lag Function

A-1

References: See p. 27

Abstract: This memorandum provides a scheme for the numerical solution of the "Lag Function," a complex function of a parameter k. The solution involves the calculation of two double integrals of infinite extent, highly oscillatory, and with singularities at the end points.

27 March 1981
GHP:SJG:lhz

TABLE OF CONTENTS

Preliminary Remarks

This report can be said to have two parts. The first part is an elucidation of some notions and principles assembled by the senior writer to expedite the calculation of the lag function.

The second part consists of detailed instructions on how to program the concepts presented in the first part. It was written by the senior writer with a view to expediting the understanding of an unknown program writer of unknown capacities. As a consequence, it is perhaps too elementary in character to appear here. However, an instructional precept of some merit is that it is better to over- rather than under-simplify. Time loss to the knowledgeable should be negligible if scanning is used judiciously.

Background Material

The term "lag function" refers to a complex function of k, which when written out in full is [1]*

$$B(k) = -\frac{u_o}{2\pi} 2k\, L(2k) \left\{ \frac{\pi}{2} + \int_1^\infty \sqrt{\frac{t}{t-1}}\, e^{-2jkt^2} dt \int_0^1 \frac{\sqrt{q(1-q)}}{q-t}\, e^{2jkq^2} dq \right.$$

$$\left. + \int_0^\infty \sqrt{\frac{t}{t+1}}\, e^{-2jkt^2} dt \int_0^1 \frac{\sqrt{q(1-q)}}{q+t}\, e^{2jkq^2} dq \right\} .$$

The development of algorithms for computing the double integrals within the braces is the problem of interest here.

---

*Numbers in brackets designate references at end of paper.

27 March 1981
GHP:SJG:lhz

The j in the exponential function is the imaginary unit and, since

$$e^{x+iy} = e^{x} \cos y + i\, e^{x} \sin y \quad ,$$

one has

$$e^{2jk(q^2-t^2)} = \cos 2k(q^2 - t^2) + i \sin 2k(q^2 - t^2)$$

$$= \cos 2kq^2 \cos 2kt^2 + \sin 2kq^2 \sin 2kt^2$$

$$+ i[\sin 2kq^2 \cos 2kt^2 - \cos 2kq^2 \sin 2kt^2] \quad .$$

Complex arithmetic is ruled out of the calculation in order to preserve desirable attributes of the expanded form of $e^{2jk(q^2-t^2)}$ so that the two integrals become eight integrals with each integrand containing a trigonometric product as a factor.

Let the q-axis be vertical and the t-axis horizontal. Then each of the products will be seen to vanish on a set of horizontal lines and a set of vertical lines that subdivide the two areas of integration, infinite strips bounded by $q = 0$ and $q = 1$, into rectangles in which the trigonometric factor, moving either horizontally or vertically, alternates in sign. As a consequence, if the integrals are considered as composed of the sum of the integrals on the rectangles, then each double integral is equivalent to the sum of the sums of a finite set of alternating series infinite in the t-direction. It follows that, if each integral is evaluated by the rectangles in a row, row by row, the question of "far enough" becomes the question of far enough for the infinite series. Since the algebraic component in the integrand of each integral does not change sign in the area of integration, the series are strictly alternating and so the far-enough question is easily answered.

The problem of evaluating the lag function thus reduces to evaluating a number of double integrals, each with an integrand well behaved in the rectangle of definition. Evaluating a multiple integral by iteration, δ has a cyclical character. Algorithms with cyclical properties tend, in program form, to be relatively simple, which suggests that iteration is the method of choice. To write a program in iterated form for a multiple integral, one first chooses a one-dimensional quadrature. This quadrature is then expanded to two (or more) dimensions by writing its quantities, except for an obvious few, with one additional independent variable, the additional variable indicating a level of integration in the iteration.

Controlling Error in Numerical Integration

As anyone at all experienced in the art of computing well knows, a prime difficulty in the computing of tables is the problem of keeping the incidence of unacceptable errors sufficiently small, for errors seem to creep in with an inevitability that sometimes seems to suggest the guidance of a malign hand. The true cause, of course, is the many points of possible entry, ranging from faulty computing procedures to a simple inversion of digits in those inputs and outputs handled by a human hand. The best of all checks, one seldom available, is the existence of a relationship between the final outputs which is known to hold. Unfortunately, no such check seems to exist for the final outputs of the lag function except the one always possible for results in tabular form, namely, finite differencing. The values given for the lag function were subjected to this test, although the test's efficacy was diminished some by the necessity of using mostly divided differences rather than ordinary. Divided differences tend not to indicate a faulty entry as precisely as ordinary differences do.

To make up for the lack of a dependable final check, considerable care was taken to ensure that the accuracy of each of the individual numerical integrations was good to three, if not four, significant figures. In general, there is no means of estimating an error in an evaluation by a particular quadrature formula other than comparing it with another evaluation using another quadrature differing in some significant respect. Some comparisons, however, do not always give reliable estimates because of a cancellation of errors.

A favored procedure for numerical evaluation of a definite integral begins by dividing the interval of integration into subintervals. The subdividing is likely to be guided to some degree by the perceived character of the integrand and by the choice of the quadrature formula which is to be used in the subintervals. The chosen quadrature obtains from the set of subintervals, a set of subintegrals which when summed is taken as a first approximation. A second approximation is then formed by using the same quadrature on the set of subintervals that come from halving the lengths of the subintervals in the first set. The two evaluations are compared by taking their difference. If the difference is sufficiently small, the second approximation is then accepted as final.

Suppose the quadrature chosen is one of the Newton-Cotes kind, say, Simpson's rule. Consider the subapproximation obtained on one subinterval of the first set. It can be written in the form

$$\frac{h}{6}\,(y_0 + 4y_2 + y_4) \quad ,$$

where h is the interval length and y_0 and y_4 are ordinates at its end points and y_2 at its midpoint. The interval length for the matching two subintervals of the second set is h/2 and their quadrature formula becomes

$$\frac{h}{12}\,(y_0 + 4y_1 + 2y_2 + 4y_3 + y_4) \quad .$$

The difference between the two subresults will be seen as h multiplied into a fourth order finite difference: $h\Delta^4 y/12$.

That the difference between the two quadratures is found to be a constant multiple of a fourth order finite difference is not entirely happenstance. For, as will be shown, the difference between any two quadratures on the same interval is always a linear combination of finite differences of the order of the lower of the two quadratures. In the case just considered, both quadratures are of the fourth order, and accordingly, their difference must be a linear combination of fourth order finite differences, which it is, though it consists of only one term.

A more interesting example is had in the difference between Simpson's rule and the four term Newton-Cotes formula:

$$h(y_0 + 3y_1 + 3y_2 + y_3)/8 \quad .$$

To have the seven abscissas of the two formulas fall evenly in a common interval, it is necessary to subdivide the interval (0,h) into six equal parts. The seven abscissas are then given by the relation

$$x_k = kh/6,\ k = 0,1,2,\ldots 6 \quad .$$

It follows that Simpson's rule takes the form

$$h(y_0 + 4y_3 + y_6)/6 \quad ,$$

the four term formula, the form

$$h(y_0 + 3y_2 + 3y_4 + y_6)/8$$

and the difference may be written as

$$h(y_6 - 9y_4 + 16y_3 - 9y_2 + y_0)/48 \quad .$$

The parenthetical quantity with y_5 and y_1, added and subtracted, is equivalent to

$$(y_6 - y_5) + (y_5 - y_4) - 8(y_4 - y_3) + 8(y_3 - y_2) + (y_2 - y_1) + (y_1 - y_0)$$

or, in terms of first order finite differences, to

$$\Delta y_5 + \Delta y_4 - 8\Delta y_3 + 8\Delta y_2 - \Delta y_1 - \Delta y_0 \quad .$$

This conversion of a linear combination of finite differences of order zero to one of the first order is accomplished by grouping. Additional groupings, three in number, reduce the initial linear combination to one of the fourth order*: $\Delta^4 y_2 + 4\Delta^4 y_1 + \Delta^4 y_0$,

*The same result can be obtained more easily by a parallel to synthetic division. Consider the following array which is like that of synthetic division.

$$\begin{array}{rrrrrrrl}
1 & 0 & -9 & 16 & -9 & 0 & 1 & \vert\underline{1} \\
 & 1 & 1 & -8 & 8 & -1 & -1 & \\
\hline
1 & 1 & -8 & 8 & -1 & -1 & \vert\underline{1} & \\
 & 1 & 2 & -6 & 2 & 1 & & \\
\hline
1 & 2 & -6 & 2 & 1 & \vert\underline{1} & & \\
 & 1 & 3 & -3 & -1 & & & \\
\hline
1 & 3 & -3 & -1 & \vert\underline{1} & & & \\
 & 1 & 4 & 1 & & & & \\
\hline
1 & 4 & 1 & & & & &
\end{array}$$

The first course of numbers in this display are the numerical coefficients in the linear combination of finite differences of order zero; the second, the coefficients in the linear combination of first order differences and so on, course by course. The last fully interpreted is

$$\Delta^4 y_2 + 4\Delta^4 y_1 + \Delta^4 y_0 \quad .$$

---

a linear combination of the kind designated by the principle stated above. The full expression for the difference between the two original quadratures becomes

$$h(\Delta^4 y_2 + \Delta^4 y_1 + \Delta^4 y_0)/48 \quad .$$

The principle gives only the order of the finite differences in the final linear combinations. It does not give the coefficients or the total number of the differences. The latter quantity, however, can be deduced from the total number of abscissas in the difference of the quadratures. The number of abscissas involved in the case using Simpson's rule is five. On five points only one fourth order finite difference is possible. In the second case with the addition of abscissas x_2 and x_4, the total is seven. Without going through the whole procedure, one knows that only three finite fourth order differences are possible. So all that is learned from the full procedure is that the multipliers (1,4,1), each divided by 48, will appear in the final result.

Both error estimates in the two illustrative examples are combinations of ordinary fourth order finite differences. So, if the integrands in the subintervals have small variation, the individual errors, and also the sum

sum of their absolute values over the subintervals, can be expected to diminish with some rapidity as the subintervals are halved. In the instance of the second example, however, cancellation between the three finite differences may indicate an error appreciably less than actual and allow acceptance of an incorrect result. This may seem very improbable but the writer's experience suggests that, when it comes to errors, Murphy's law always should be remembered.

The Newton-Cotes formulas with their even spacing often have a desirable simplicity in some applications of numerical integration. They are, as is well known, much less powerful than those of the Gaussian kind. The calculation of the lag function, though not awe inspiring by today's standards, is still large enough to warrant using Gaussian quadratures.

The principle, whose validity was established in two instances with quadratures of the Newton-Cotes kind also holds for the Gaussian, the Radau, the Lobatto quadratures and others, but because of the uneven spacing of their abscissas, ordinary finite differences must be replaced by divided. Divided differences do not yield so readily to manipulation. A proof of general validity must be found.

One illustrative example tells in essence the whole story. Suppose the two quadratures are the two term and the three term Gaussian quadratures on the same interval. Let the two quadratures be represented by the quantities

$$c_1y_1 + c_3y_3$$

and

$$c_0y_0 + c_2y_2 + c_4y_4 \quad ,$$

where the c's are weights and the y's ordinates. Consider their difference

$$c_0y_0 - c_1y_1 + c_2y_2 - c_3y_3 + c_4y_4 \quad .$$

The two term Gaussian gives the exact value of the integral of any polynomial of third degree, the three term Gaussian, of any polynomial of the fifth degree. Let the polynomials be the first four powers of x, namely, 1, x, x^2 and x^3. Both formulas give the exact value of the integrals for these powers of x. When each power is used in succession to supply values for the ordinates in the difference, one obtains a system of four homogeneous equations in the c's:

$$\sum_{j=0}^{4} c_j x_j^k = 0 \quad , \quad k = 0,1,\ldots,4 \quad .$$

The rank of the matrix of the system is three. Therefore, a solution of the four-equation system exists, other than $c_0 = c_1 = c_2 = c_3 = c_4$. Any constant multiple of this solution is also a solution. The fourth order finite difference on the five points (x_0,y_0), (x_1,y_1), (x_2,y_2), (x_3,y_3) and (x_4,y_4) is also a linear combination of ordinates and also must vanish for the four powers of x. In view of the generality of this last proof of the principle stated above, it must be true for any two quadrature formulas.

27 March 1981
GHP:SJG:lhz

A Succession of Explanatory Algorithms Preliminary to the Working Algorithms

Although the senior author, over the years, has formulated a number of algorithms for expression and calculation in the language of FORTRAN, he has never written a program in the language, only through the services of an intermediary. Illiteracy in FORTRAN and, for that matter, all computer languages persisted until the appearance of JOSS*, a language so obvious and consistent in terminology and forms that to learn it is a negligible task and to write a program in it, a pleasure.

In the case of the lag function, several algorithms for approximating its definite integrals were written in JOSS and checked for faults. All use basic Gaussian quadratures: one of two terms, one of three, and one of four. The K-th abscissa of the quadrature of J terms is represented by D(J,K) and the K-th weight by C(P,K). For the interval (0,1), the values of the C's are:

$C(2,1) = .5$, $C(2,2) = C(2,1)$
$C(3,1) = .2777\ldots$, $C(3,2) = .444\ldots$,
$C(3,3) = C(3,1)$
$C(4,2) = .326072577431273$, $C(4,1) = .173927422568727$,
$C(4,3) = C(4,2)$ $C(4,4) = C(4,1)$

and the values of the D's are:

$D(2,1) = .5 - \sqrt{3}/6$, $D(2,2) = 1 - D(2,1)$,
$D(3,1) = .112701665379258$, $D(3,2) = .5$,
$D(3,3) = 1 - D(3,1)$,
$D(4,1) = .069431844202973$, $D(4,2) = .330009478207572$,
$D(4,3) = 1 - D(4,2)$, $D(4,4) = 1 - (4,1)$

*An acronym for "Johnniac Open Shop System".

The first of the explanatory algorithms, explanatory with regard to JOSS and to the working algorithms, follows:

Algorithm 1*

```
1.12  Set H = B - A
1.18  Set J = 1
1.3   Set V(J) = 0
1.38  Set K = 0
1.4   Set J = J + 1
1.42  Set V(J) = 0
1.5   Set K = K + 1
1.52  Set X = A + D(J,K)*H
1.56  Set V(J) = V(J) + V(J,K)*H*F(X)
1.58  To Step 1.48 if K < J
1.62  To Step 1.94 if |V(J) - V(J-1)| ≦ E
1.64  To Step 1.38 if J < 4
1.94  Type V(J).
```

This program, which is capable of approximating the definite integral of F(X) on the interval (A,B), exhibits several essential characteristics of the JOSS language: Programs are written in parts, where a part is similar to a FORTRAN subroutine with a return. A part is made up of a succession of numbered commands. The number on a command is the sum of an integer, which does not differ for commands from the same part and a decimal fraction. The commands are grammatical to the extent that the first letter of the first word, always a verb in the imperative mood**, is capitalized and the end of the command is signaled by a period.

The flow of the program is easily grasped. The body of the program implements the three quadratures one by one in a cyclical fashion. The quantity E in Step 1.62 specifies an allowable error and, barring the

*See Appendix A for FORTRAN equivalent.

**The verb "go" in Steps 1.58, 1.62 and 1.64.

unusual, if V(J) is accepted for J = 2, 3 or 4, the error in the approximation obtained is presumably less than or equal to E. If V(2) is accepted, it follows, because Step 1.3 sets V(1) to zero, that the approximation is less than E.

When V(4) is not accepted by Step 1.62, the control is not sent to Step 1.38, but passes on to Step 1.94 for a print out. Inasmuch as the quadrature of four terms yields the correct result for a polynomial integrand of the seventh degree, the small program may well be adequate for the integrands it will be called on to evaluate. After all, the integrands it needs to approximate do not necessarily behave in an outrageous fashion. To cover the contingency that some will be difficult, the small program is so modified in Algorithm 2 (which follows) that if V(4) is not accepted, the interval (A,B) is halved and each half attacked separately.

If the interval of integration is to be halved and rehalved, there are many ways to go about it. One way is presented in the following algorithm:

Algorithm 2*

```
1.1   Set O = 1
1.12  Set H(O) = B - A
1.14  Set U(O) = A
1.16  Set S = 0
1.18  Set J = 1
1.3   Set V(J) = 0
1.38  Set J = J + 1
1.4   Set K = 0
1.42  Set V(J) = 0
1.5   Set K = K + 1
1.52  Set X = U(O) + D(J,K)*H(O)
1.56  Set V(J) = V(J) + C(J,K)*H(O)*F(X)
1.58  To Step 1.5 if K < J
1.62  To Step 1.18 if |V(J) - V(J-1)| ≦ E
1.64  To Step 1.38 if J < 4
```

*See Appendix A for FORTRAN equivalent.

```
1.66  H(0)     = H(0)/2
1.68  Set 0    = 0 + 1
1.7   Set H(0) = H(0-1)
1.72  Set U(0) = U(0-1) + H(0)
1.76  To Step 1.18
1.84  Set S    = S + V(J)
1.86  To Step 1.94 if 0 = 1
1.88  Set 0 = 0 - 1
1.9   To Step 1.18
1.94  Type S.
```

In subdividing an interval, the symbol 0 is introduced whose largest value at any time is equal to the number of intervals available to the algorithm for processing. All intervals are tagged with an 0-value. At the beginning there is only one interval, [A,B]. So 0 = 1. If the three quadratures are unable to reach an acceptable result on this interval, it is divided into two equal parts, the part nearest B is given the value 0 = 1, the other part retains the value 1. The attack shifts to the interval with 0 = 1. If the attack is successful, the attack moves to 0 = 1. If unsuccessful, 0 = 2 is divided and the two parts tagged with the cardinal numbers 0 = 2 and 0 = 3. Always the interval with the highest 0-value is the one in process until ultimately success comes to the interval 0 = 1.

Algorithm 2, it will be noticed, is flawed with respect to the control of error. Subdivision of intervals leads to an accumulation of error that may exceed the value of E specified in Step 1.62. Although the algorithm can be modified so as to insure an error less than E, the simplest and a likely adequate modification is a carefully chosen decrease in the value of E, particularly in view of the fact that there are a number of other sources of error to protect against on the way to the value of B(k).

The second algorithm will now be used to illustrate the conversion of an integration in one dimension to an iterated integral in two dimensions.

Algorithm 3*

```
1.1   Set O(I) = 1
1.12  Set H[I,O(I)] = B(I) - A(I)
1.14  Set U[I,O(I)] = A(I)
1.16  Set S(I) = 0
1.18  Set J(I) = 1
1.3   Set V[I,J(I)] = 0
1.38  Set J(I) = J(I) + 1
1.4   Set K(I) = 0
1.42  Set V[I,J(I)] = 0
1.5   Set K(I) = K(I) + 1
1.52  Set X(I) = U[I,O(I)] + D[J(I),K(I)]*H[I,O(I)]
1.54  Do Part 2 for I = 2 if I = 1
1.56  Set V[I,J(I)] = V[I,J(I)] + C[J(I),K(I)]*H[I,O(I)]*G(I)
1.58  To Step 1.5 if K(I) < J(I)
1.62  To Step 1.84 if |V[I,J(I)] - V[I,J(I) - 1]| ≦ E(I)
1.64  To Step 1.38 if J(I) < 4
1.66  Set H[I,O(I)] = H[I,O(I)]/2
1.68  Set O(I) = O(I) + 1
1.7   Set H[I,O(I)] = H[I,O(I) - 1]
1.72  Set U[I,O(I)] = U[I,O(I) - 1] + H[I,O(I)]
1.76  To Step 1.18
1.84  Set S(I) = S(I) + V[I,J(I)]
1.86  To Step 1.94 if O(I) = 1
1.88  Set O(I) = O(I) - 1
1.9   To Step 1.18
1.94  Type S(1) if I = 1

2.3   Do Part 1
2.7   Set I = 1
```

Except for the index I, the last algorithm and the preceding one are identical for the first 11 steps. Step 1.54 has no counterpart. The counterpart, if it existed, would fall between Steps 11 and 12. Step 12 and those following closely parallel Step 1.56 and following. The function G(I) of Step 1.56 is defined by the relations

$$G(I) = \begin{cases} S(2) & , \text{ if } I = 1 \ , \\ F[X(1),X(2)] & , \text{ if } I = 2 \ . \end{cases}$$

*See Appendix A for FORTRAN equivalent.

The value of G(I) for I = 2, identifies the iterated interal as

$$\int_{A(1)}^{B(1)} dx \int_{A(2)}^{B(2)} F(x,y)\, dy \quad .$$

It helps in understanding the two-dimensional algorithm to call to mind the principles of the iterated integral. As an example, take the iterated integral displayed above. The last integration is a definite integral with respect to x having as its integrand the result of a definite integral with respect to y the latter having F(x,y) as an integrand with x treated as a constant. If the steps in the algorithm are examined carefully, it is seen that the algorithm in its own way follows the same path. The algorithm starts with a quadrature with respect to x, or rather X(1). The first abscissa is the first abscissa of the two-point Gaussian quadrature. But G(I) in Step 1.56 requires S(2), which is the result of a quadrature with respect to y [or X(2)] with the integrand F[X(1),X(2)], the variable X(1) being held constant. Hence, the quadrature with respect to X(1) must be temporarily abandoned. This is accomplished by Step 1.54 which sends the control to Part 2 with I = 2. The first step of Part 2, Step 2.3, sends the control back to Part 1, i.e., Step 1.1. The quadrature with respect to X(2) then begins and continues without interruption until completed. Control then returns to Step 2.7, which sets I = 1, completing Part 2. Control returns to Step 1.56 with the required values of S(2) now available. Hence the quadrature with respect to X(1) can continue until shortly Step 1.54 is encountered again, and so on.

E(1) and E(2) should not be given the same value because the integrand for the X(1)-quadrature has a component from the X(1)-quadrature and so many have a scatter as large as E(2). Finite differences are well known for their sensitivity to scatter. Consequently, the programmed check for the X(1)-quadrature tends to magnify the apparent estimate, making acceptance difficult, or even impossible. Experience indicates that E(1) and E(2) should satisfy the relation,

$$E(1) \leqslant E(2)/10 \quad .$$

The Algorithm for the Integral

$$\int_0^\infty \sin 2kt^2 \sqrt{\frac{t}{t+1}}\, dt \int_0^1 \sin 2kq^2 \frac{\sqrt{q-q^2}}{q+t}\, dq \quad .$$

The second integral in the definition of B(k), unlike the first, is free of such nuisances as discontinuities in its integrand and, if one selects $\sin 2kt^2 \sin 2kq^2$ from the four components of $e^{2jk(q^2-t^2)}$ as the trigonometric factor, one has the simplest of the eight that are implicit in B(k)'s definition. The quantity $\sin 2kx^2$ vanishes when $x = \sqrt{n\pi/(2k)}$ and $n = 0,1,2,\ldots$. Hence the area of integration is divided into rectangular sub-areas by the lines

$$q = \sqrt{n^*p} \quad ,$$

$$t = \sqrt{m^*p} \quad ,$$

where $p = \pi/(2k)$, $n = 0,1,2,\ldots$ and $m = 0,1,2,\ldots$. The integration over the infinite strip is accomplished in a piecewise fashion. Starting with the rectangle common to the first row and first column, the calculation proceeds rectangle-by-rectangle along the row until a suitable test indicates the remainder for the infinite alternating series is sufficiently small. The next row is then given the same treatment and finally the finite series of sums is summed to give the value of the sin $2kt^2$ sin $2kq^2$ component of the second integral. The steps in the algorithm, their explanation delayed, follow.

```
1.1   Set p = π/(2*k)
1.16  Set N(2) = 1
1.18  Set A(2) = 0
1.2   Set N(1) = 1
1.22  Set A(1) = 0
1.24  Set T[N(2)] = 0
1.26  Set B(2) = sqrt[N(2)*p]
1.28  To Step 1.32 if B(2) < 1
1.3   B(2) = 1
1.32  Set B(1) = sqrt[N(1)*p]
1.38  Do Part 2 for I = 1
1.42  Set T[N(2)] = T[N(2)] + S(1)
1.44  To Step 1.56 if N(1) < i(1)
1.46  To Step 1.56 if |S(1)| > E(3)
1.48  To Step 1.56 if s*S(1) > 0
1.5   Set Z(1) = Z(1) + 1
1.52  To Step 1.66 if Z(1) > i(1)
1.54  To Step 1.58
1.56  Set Z(1) = 0
1.58  Set s = S(1)
1.60  Set A(1) = B(1)
1.62  Set N(1) = N(1) + 1
1.64  To Step 1.32
1.66  To Step 1.88 if B(2) = 1
1.68  To Step 1.8 if N(2) < i(2)
1.7   To Step 1.8 if |T[N(2)]| > E(3)
1.72  To Step 1.8 if T[N(2)]*T[N(2) - 1] > 0
1.74  Set Z(2) = Z(2) + 1
1.76  To Step 1.88 if Z(2) > i(2)
1.78  To Step 1.82
1.8   Set Z(2) = 0
1.82  Set A(2) = B(2)
1.84  Set N(2) = N(2) + 1
1.86  To Step 1.2
1.88  Print output

2.1   Set O(1) = 1
2.12  Set H[I,O(I)] = B(I) - A(I)
2.14  Set U[I,O(I)] = A(I)
2.16  Set S(I) = 0
2.18  Set J(I) = 1
2.3   Set V[I,J(I)] = 0
2.38  Set J(I) = J(I) + 1
```

```
2.4    Set K(I) = 1
2.42   Set V[I,J(I)] = 0
2.5    Set K(I) = K(I) + 1
2.52   Set X(I) = U[I,O(I)] + D[J(I),K(I)]*H[I,O(I)]
2.54   Do Part 3 for I = 2 if I = 1
2.56   Set V[I,J(I)] = V[I,J(I)] + C[J(I),K(I)]*H[I,O(I)]*G(I)
2.58   To Step 2.5 if K(I) < J(I)
2.62   To Step 2.84 if |V[I,J(I)] - V[I,J(I) - 1]| ≤ E(I)
2.66   Set H[I,O(I)] = H[I,O(I)]/2
2.68   Set O(I) = O(I) + 1
2.7    Set H[I,O(I)] = H[I,O(I) - 1]
2.72   Set U[I,O(I)] = U[I,O(I) - 1] + H[I,O(I)]
2.76   To Step 2.18
2.8    Set S(I) = S(I) + V[I,J(I)]
2.86   Done if O(I) = 1
2.88   Set O(I) = O(I) - 1
2.9    To Step 2.18

3.3    Do Part 2
3.7    I = 1
```

Inputs for the algorithm are, of course, k, the weights and abscissas for the three Gaussian quadratures, and the critical quantities E(1) and E(2). In addition, there is a third critical quantity, E(3), and two critical integers i(1) and i(2). Their roles are made clear in the following explanation of the steps of the algorithm.

Part 2 carries the main body of the computation and can be said to be served by Parts 1 and 3. Steps prior to Step 1.38 prepare the way for Part 2, Step 1.38 initiates the computations of Part 2 and, when Part 2 has finished with a rectangle, steps subsequent to Step 1.38 sum the result, select the next rectangle, if another is required, and take over the functions of some of the steps prior to Step 1.38.

The quantities N(1) and N(2) in Steps 1.16 and 1.18 are defined by stating that the number pair [N(1),N(2)] are the referene numbers for the rectangle common to the N(1)th column and N(2)th row.

T[N(2)] of Step 1.24 is the running total, accumlated rectangle-by-rectangle along the N(2)th row.

Steps 1.44 through 1.56 express properties of convergent alternating series. It will be recalled that when a series reaches a term, after which it is strictly alternating and the magnitude of each term is less than the preceding term, then the magnitude of the last term calculated is a bound on the remainder. The quantity Z(1) in Steps 1.5, 1.52 and 1.56 is a count of the number of times in succession the terms from a row have met these conditions. Step 1.44 does not allow the count to start until N(1) reaches a value thought to be large enough to make acceptance by happenstance unlikely. The character of the integrand insures that such a value exists. Step 1.46 returns the count to zero, whenever S(1), the magnitude of the N(1)th term, is not less than an appropriately chosen critical quantity. Step 1.48 sets the count to zero, if two successive terms (s is the value of the preceding term) are of the same sign. When the count succeeds in reaching a value greater than the integer i(1), also used in Step 1.44*, the process is presumed to have shown that further summing of the series is unnecessary and control passes to Step 1.66. If, however, the count does not exceed i(1), Step 1.54 sends the control to Steps 1.58 through 1.64 in which the first three steps are partial preparation for the next rectangle in the row before Step 1.64 passes the control to Step 1.32, where the preparation is completed.

---

*Obviously, if experience indicates differing values should be used, there is no reason why not.

The Algorithm for the cos $2kt^2$ cos $2kq^2$ Component of the Second Integral

Because cos $2kx^2$ vanishes for $x = \pi/2, 3\pi/2, 5\pi/2, \ldots$, the width of the first row and also the first column is $\pi/(4k)$ and the lines subdividing the area of integration are

$$t = \sqrt{(2n-1)*p}$$

and

$$q = \sqrt{(2m-1)*p} \quad ,$$

where $p = \pi/(4*k)$, $(2n-1) = 1,3,5,\ldots$ and $(2m-1) = 1,3,5,\ldots$.

Only a few steps in the algorithm for the sin $2kt^2$ sin skq^2 component change. The steps are:

```
1.1    Set p      = $\pi/(4*k)$
1.26   Set B(2)   = sqrt [(2*N(2) - 1)*p]
1.32   Set B(1)   = sqrt [(2*N(1) - 1)*p]
```

The Algorithm for the sin $2kt^2$ cos $2kq^2$ Component of the Second Integral

For this component the width of the first column is $\pi/(2k)$ and of the first row $\pi/(4k)$. Two p-values are needed

$$p(1) = \pi/(2*k)$$

and

$$p(2) = \pi/(4*k) \quad .$$

Hence

```
1.1    Set p(1) = $\pi/(2*k)$
1.12   Set p(2) = $\pi/(4*k)$
1.26   Set B(2) = sqrt [(2*N(2) - 1)*p(2)]
1.32   Set B(1) = sqrt [N(1)*p(1)]
```

The Algorithm for the cos $2kt^2$ sin $2kq^2$ Component of the Second Integral

In this case

$$p(1) = \pi/(4k) \quad ,$$

and

$$p(2) = \pi/(2k) \quad ,$$

so

```
1.1   Set p(1) = π/(4*k)
1.12  Set p(2) = π/(2*k)
1.26  Set B(2) = sqrt [N(2)*p(2)]
1.32  Set B(1) = sqrt [(2*N(1) - 1)*p(1)]
```

The Algorithms for the First Integral

The first integral differs from the second in its area of integration and its integrand. The area of integration of the former is a strip in the t-direction with a finite boundary at t = 1. The area of integration of the latter is the same except that the finite boundary is at t = 0. This difference is of no great consequence.

The difference in the integrands is a little more significant. The first integral has the quantities $\sqrt{t - 1}$ and $q - t$ in its integrand, the second, the quantitites $\sqrt{t + 1}$ and $q + t$.

The infinite singularity originating in $\sqrt{t - 1}$ is integrable and readily handled by means of quadrature formulas for the weight function $1/\sqrt{t}$.

The singularity caused by $q - t$ is not integrable, but it appears in the corner of the area of integration, where $q = t = 1$ and where the quantity $\sqrt{q(1 - q)}$ vanishes. The improper double integral can be shown to exist by comparing it with the improper double integral obtained by replacing the trigonometric factor by unity. The singularity is no problem numerically, though it does slow the evaluation.

The changes necessary to convert algorithms for the second integral into those for the first are few. In the case where the trigonometric factor is sin $2kt^2$ sin $2kq^2$ they are:

```
1.12  Set n = first [i = 1(1)100 : i*p > 1]
1.2   Set X(1) = n
1.22  Set A(1) = 1
2.51  To Step 2.53 if I = 1 and U[I,O(I)] = 1
2.525 To Step 2.54
2.53  Set X(I) = U[I,O(I)] + d[J(I),K(I)]*H[I,O(I)]
2.55  To Step 2.57 if I = 1 and U[I,O(I)] = 1
2.565 To Step 2.58
2.57  Set V[I,J(I)] = V[I,J(I)] = c[J(I),K(I)] * sqrt (H[I,O(I)]  * G(I)
```

The right-hand member of Step 1.12 is a function of i that finds the first of a sequence of values of i that meets the prescribed condition. In this case, the sequence is i = 1,2,3,...,100 and the prescribed condition is that p * i be greater than unity. The value of n for each value of k can, of course, be found by a brief calculation and treated as an input. One can, however, elect to write a small subroutine equivalent to the function of Step 1.12.

The steps added to Part 2, it will be seen, provide an alternative to Gaussian quadratures whenever I = 1 and U[I,O(I)] = 1. The alternative is a Gaussian quadrature for the weight function $1/\sqrt{t}$. The lower case symbols, c and d, are respectively the weights and abscissas corresponding to the latter weight function on the interval [0,1]. They are derivable from the weight and abscissas for w(x) = 1 in the interval (-1,1). See Appendix B.

The changes necessary to convert the remaining three algorithms for the second integral into those for the first integral differ very little from those above.

The steps added to Part 2 are still needed to take care of the singularity at $t = 1$ and must, of course, be retained. The only step that changes in Part 1 is Step 1.12. For the factors $\sin 2kt^2$, $\cos 2kq^2$, one has 1.12 n = first [i = 1(1)100: i*p(1) > 1]. The only change is the unit index, or subscript, for p. For the factors $\cos 2kt^2$, $\cos 2kq^2$ and $\cos 2kt^2$, $\sin 2kq^2$ the step defining n becomes

1.12 n = first [i = 1(1)100: (2*i - 1)*p(1) > 1].

Step 1.12 now reads n = the first value of i in the sequency i = 1,2,3,...,100 such that (2*i - 1)*p(1) > 1 and n = 1,3,5,...,199.

The complete FORTRAN program for computing the lag function is presented in Appendix C.

References

[1] Parkin, B. R., "Fully Cavitating Hydrofoil Response to Streamwise Sinusoidal and Sharp-Edged Gusts at Zero Cavitation Number," ARL/PSU TM 74-172, Applied Research Laboratory, The Pennsylvania State University (4 June 1974).

[2] Atkinson, K. E., An Introduction to Numerical Analysis, John Wiley and Sons, Inc., pp. 213-279 (1977).

[3] Abramowitz, M. and I. A. Stegun, Handbook of Mathematical Functions, National Bureau of Standards, 9th Printing (November 1970).

[4] Szego, G., "Orthogonal Polynomials," Am. Math. Pub., Vol. XXIII.

Appendix A: FORTRAN Algorithms

The following algorithms are the FORTRAN coded equivalents of the first three algorithms appearing in the text. Statement numbers occurring in the text have been retained, where possible, to facilitate easy following of the logical path of the programs.

Algorithm 3 has been written as two subroutines. The first performs the "outer" iteration which depends on the value returned by the second or "inner" iteration. The subroutines would, in practice, be called by a main program which passes the values of the "outer" limits of integration [A,B], the "inner" limits [A2,B2] and a parameter R {if needed for the evaluation of F(X,Y,R)}. The result of the double integration is the variable S returned to the main program.

ALGORITHM 1: 2, 3, 4 POINT GAUSSIAN QUADRATURE ON FULL INTERVAL [A, B]

```
      PROGRAM GAUSS1
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION C(4,4),D(4,4),V(4)
      DATA E/.0002/,A/0.0/,B/3.14159265/,C(2,1)/.5D0/,D(3,2)/.5D0/
      DATA C(3,1)/.277777777777777D0/,D(4,1)/.069431844202973D0/
      DATA C(3,2)/.444444444444444D0/,C(4,2)/.326072577431273D0/
      DATA C(4,1)/.173927422568727D0/,D(3,1)/.112701665379258D0/
      DATA D(4,2)/.330009478207572D0/,IPR/3/
      F(X)=DSIN(X)
      C(2,2)=C(2,1)
      C(3,3)=C(3,1)
      C(4,3)=C(4,2)
      C(4,4)=C(4,1)
      D(2,1)=.5D0-DSQRT(3.0D0)/6.0D0
      D(2,2)=1.0D0-D(2,1)
      D(3,3)=1.0D0-D(3,1)
      D(4,3)=1.0D0-D(4,2)
      D(4,4)=1.0D0-D(4,1)
      H=B-A
      V(1)=0.0D0
      DO 5 J=2,4
      V(J)=0.0D0
      DO 7 K=1,J
      X=A+D(J,K)*H
7     V(J)=V(J)+C(J,K)*H*F(X)
      IF(DABS(V(J)-V(J-1)).LE.E) GO TO 13
5     CONTINUE
13    WRITE(IPR,1) J,V(J)
1     FORMAT(I2,2X,E16.8)
      STOP
      END
```

ALGORITHM 2: 2, 3, 4 POINT GAUSSIAN QUADRATURE WITH SUBDIVISION OF INTERVAL [A, B]

```
      PROGRAM GAUSS2
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION C(4,4),D(4,4),V(4),H(10),U(10)
      DATA E/.0002/,A/0.0/,B/3.14159265/,C(2,1)/.5D0/,D(3,2)/.5D0/
      DATA C(3,1)/.27777777777777770D0/,D(4,1)/.06943184420297300/
      DATA C(3,2)/.44444444444444440D0/,C(4,2)/.32607257743127300/
      DATA C(4,1)/.17392742256872700/,D(3,1)/.11270166537925800/
      DATA D(4,2)/.33000947820757200/,IPR/3/
      F(X)=DSIN(X)
      C(2,2)=C(2,1)
      C(3,3)=C(3,1)
      C(4,3)=C(4,2)
      C(4,4)=C(4,1)
      D(2,1)=.5D0-DSQRT(3.0D0)/6.0D0
      D(2,2)=1.0D0-D(2,1)
      D(3,3)=1.0D0-D(3,1)
      D(4,3)=1.0D0-D(4,2)
      D(4,4)=1.0D0-D(4,1)
      N=1
      H(N)=B-A
      U(N)=A
      S=0.0D0
5     V(1)=0.0D0
      DO 7 J=2,4
      V(J)=0.0D0
      DO 10 K=1,J
      X=U(N)+D(J,K)*H(N)
10    V(J)=V(J)+C(J,K)*H(N)*F(X)
      IF(DABS(V(J)-V(J-1)).LE.E) GO TO 21
7     CONTINUE
      H(N)=H(N)/2.0
      N=N+1
      H(N)=H(N-1)
      U(N)=U(N-1)+H(N)
      GO TO 5
21    S=S+V(J)
      IF(N.EQ.1) GO TO 25
      N=N-1
      GO TO 5
25    WRITE(IPR,1) J,S
1     FORMAT(I2,2X,E16.8)
      STOP
      END
```

ALGORITHM 3: TWO DIMENSIONAL 2, 3, 4 POINT GAUSSIAN QUADRATURE WITH SUBDIVISION OF INTERVAL [A,B]

```
      SUBROUTINE GAUSSA(A,B,A2,B2,R,S)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION C(4,4),D(4,4),V(4),H(10),U(10)
      DATA E/.0002D0/,C(2,1)/.5D0/,D(3,2)/.5D0/
      DATA C(3,1)/.277777777777777D0/,D(4,1)/.069431844202973D0/
      DATA C(3,2)/.444444444444444D0/,C(4,2)/.326072577431273D0/
      DATA C(4,1)/.173927422568727D0/,D(3,1)/.112701665379258D0/
      DATA D(4,2)/.330009478207572D0/,IPR/3/
      F(X,Y,R)=Y*DSIN(X)
      C(2,2)=C(2,1)
      C(3,3)=C(3,1)
      C(4,3)=C(4,2)
      C(4,4)=C(4,1)
      D(2,1)=.5D0-DSQRT(3.0D0)/6.0D0
      D(2,2)=1.0D0-D(2,1)
      D(3,3)=1.0D0-D(3,1)
      D(4,3)=1.0D0-D(4,2)
      D(4,4)=1.0D0-D(4,1)
      N=1
      H(N)=B-A
      U(N)=A
      S=0.0D0
118   V(1)=0.0D0
      DO 138 J=2,4
      V(J)=0.0D0
      DO 150 K=1,J
      X=U(N)+D(J,K)*H(N)
      CALL G1(X,Y,A2,B2,R)
150   V(J)=V(J)+C(J,K)*H(N)*F(X,Y,R)
      IF(DABS(V(J)-V(J-1)).LE.E) GO TO 134
138   CONTINUE
      H(N)=H(N)/2.0D0
      N=N+1
      H(N)=H(N-1)
      U(N)=U(N-1)+H(N)
      GO TO 118
184   S=S+V(J)
      IF(N.EQ.1) GO TO 194
      N=N-1
      GO TO 118
194   RETURN
      END
```

ALGORITHM 3: TWO DIMENSIONAL 2, 3, 4 POINT GAUSSIAN QUADRATURE WITH SUBDIVISION OF INTERVAL [A,B] - continuation

```
      SUBROUTINE GI(Y,S,A,B,R)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION C(4,4),D(4,4),V(4),H(10),U(10)
      DATA E/.0002D0/,C(2,1)/.5D0/,D(3,2)/.5D0/
      DATA C(3,1)/.27777777777777D0/,D(4,1)/.06943184420297D0/
      DATA C(3,2)/.44444444444444D0/,C(4,2)/.32607257743127D0/
      DATA C(4,1)/.17392742256872D0/,D(3,1)/.11270166537925D0/
      DATA D(4,2)/.33000947820757D0/,IPR/3/
      F(X,Y,R)=DSIN(X)
      C(2,2)=C(2,1)
      C(3,3)=C(3,1)
      C(4,3)=C(4,2)
      C(4,4)=C(4,1)
      D(2,1)=.5D0-DSQRT(3.0D0)/6.0D0
      D(2,2)=1.0D0-D(2,1)
      D(3,3)=1.0D0-D(3,1)
      D(4,3)=1.0D0-D(4,2)
      D(4,4)=1.0D0-D(4,1)
      N=1
      H(N)=B-A
      U(N)=A
      S=0.0D0
118   V(1)=0.0D0
      DO 138 J=2,4
      V(J)=0.0D0
      DO 150 K=1,J
      X=U(N)+D(J,K)*H(N)
150   V(J)=V(J)+C(J,K)*H(N)*F(X,Y,R)
      IF(DABS(V(J)-V(J-1)).LE.E) GO TO 184
138   CONTINUE
      H(N)=H(N)/2.0D0
      N=N+1
      H(N)=H(N-1)
      U(N)=U(N-1)+H(N)
      GO TO 118
184   S=S+V(J)
      IF(N.EQ.1) GO TO 194
      N=N-1
      GO TO 118
194   RETURN
      END
```

Appendix B: Two Sets of Gaussian Weights and Abscissas

INTERVAL = [0,1]   WEIGHT FUNCTION = 1

W_I	X_I
n=2	
.5	.21132 48654 05187
.5	.78867 51345 94813
n=3	
.27777 77777 77778	.11270 16653 79258
.44444 44444 44444	.5
$C_3 = C_1$	.88729 83346 20742
n=4	
.17392 74225 68727	.06943 84420 29730
.32607 25774 31273	.33000 94782 07572
$C_3 = C_2$	.66999 05217 92428
$C_4 = C_1$	.93056 81557 97027
n=5	
.11846 34425 28095	.04691 00770 30668
.23931 43352 49683	.23076 53449 47159
.28444 44444 44444	.5
$C_4 = C_2$	.76923 46550 52841
$C_5 = C_1$	.95308 99229 69332
n=6	
.08566 22461 89585	.03376 52428 98424
.18038 07865 24070	.16939 53067 66867
.23395 69672 86345	.38069 04069 58401
$C_4 = C_3$	.61930 95930 41599
$C_5 = C_2$	.83060 46932 33133
$C_6 = C_1$	.96623 47571 01576

INTERVAL = [0,1] WEIGHT FUNCTION = $1/\sqrt{x}$

x_i	w_i
n=2	
.11558 71099 97048	1.3042 90309 72509
.74155 57471 45810	.69570 96902 74908
n=3	
.05693 11596 70074	.93582 78691 45382
.43719 78527 51095	.72152 31460 96278
.86908 43784 32472	.34264 89847 58340
n=4	
.03364 82680 67507	.72536 75667 56724
.27618 43138 72464	.62741 32917 55774
.63467 74762 34637	.44476 20689 06748
.92215 66084 92058	.20245 70725 80752
n=5	
.02216 35688 07218	.59104 84494 29951
.18783 15676 52445	.53853 34386 19992
.46159 73614 96266	.43817 27250 31964
.74833 46283 87281	.29890 26983 01162
.94848 39262 28836	.13334 26886 17376
n=6	
.01568 34066 07400	.49829 40916 26806
.13530 00116 55248	.46698 50730 76710
.38069 04069 58401	.40633 48534 46132
.61930 95930 41599	.32015 66570 86692
.81742 80132 66875	.21387 86519 90636
.96346 12963 70913	.09435 06727 73024

Appendix C: Implementation of Algorithms for Computing the Lag Function

The following example is an adaptation of the preceding algorithms applied to the computation of the lag function. Implementation follows directly from that outlined in the main body of this manuscript. The few exceptions are clarified in the following outline.

I. MAIN PROGRAM: For each value of R (R=k), eight double integrals are evaluated. The real and imaginary parts (SRE, SIM respectively) are the sums of the corresponding contributions (SUM(I)) of the appropriate integrals. No output statements are supplied as the relevant data are subject to user discretion.

For efficiency and economy, the q interval [0,1] was held constant in the main program since the q algorithm provides for subdivision of the interval when necessary. In light of this, a six-point formula was used which is theoretically exact for polynomials of order eleven or less. For the maximum value of k used (10.0), the number of nodes on the interval [0,1] is less than eleven.

T(1) and T(2) are the n and (n-1) partial sums of the alternating series. When the series is terminated, they provide an upper and lower bound on the solution. T(3) is the midpoint and is accepted as the correct result. T(4) and T(5) are summations of the total estimated errors in the t and q integration respectively.

E(INT) is used as a pseudo relative error bound on the n^{th} partial sum. When the contribution to T(1) is less than E(INT) or some predetermined value (.0001 in this case), the summation of the alternating series terminates.

II. SUBROUTINES:

(a) GAUSS1, GAUSS2 - These perform the quadratures for t and q respectively. The weights and abscissas for each type of quadrature are initialized in GAUSS1 and passed to GAUSS2 in the argument list of the CALL statement. These subroutines are the FORTRAN equivalents of the last algorithm in the text which uses a relative error bound. Variables not appearing in the algorithm are explained as follows.

JJ(N) saves the value of the order of the last quadrature accepted. This prevents duplication of effort in recalculating lower order quadratures previously computed and accepted by GAUSS1 and GAUSS2.

SV(N) saves the contribution of the N^{th} subinterval which is added to the total sum at Step 100 should the condition at Step 200 be met.

EQ1 is the estimated error in the q-integration for a specific value of t. TEQ is the sum of the estimated q error in the entire t interval just computed.

(b) PVAL - Calculates the factor used to determine the zeroes of the sin or cos argument for the boundaries of the subintervals on the infinite t range.

(c) LVAL - Determines the integer multiplier for the first upper limit in t when the lower limit = 1.0.

(d) AIV - Sets the lower limit on the t interval to 0.0 or 1.0 depending on which integral is being evaluated.

(e) BIV - Calculates the next node point on the t interval.

(f) FT/FQ - Evaluates the function at a specific abscissa in the appropriate t/q quadrature.

```
***** MAIN PROGRAM *****
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION T(5),SUM(8),E(8),RR(34)
      COMMON A1,B1,A2,B2,R,INT
      DATA I1/5/,RE/.0005D0/,IPR/6/,PI/3.14159265358979D0/
      DATA RR/.01,.02,.04,.06,.08,.1,.12,.15,.2,.3,.4,.5,.6,.7,.8,.9,
     >1.0,1.2,1.4,1.6,1.8,2.0,2.5,3.0,3.5,4.0,4.5,5.0,5.5,6.0,7.0,
     >8.0,9.0,10.0/
      DO 800 NR=1,34
      R=RR(NR)
      DO 900 INT=1,8
      SUM(INT)=0.0D0
      S=0.0D0
      A2=0.0D0
      CALL PVAL(P1,P2,PI)
      CALL LVAL(P1,L)
      I1=L+4
120   M=L
      CALL A1V
      T(1)=0.0D0
      T(4)=0.0D0
      T(5)=0.0D0
      B2=1.0D0
132   CALL B1V(M,P1)
      CALL GAUSS1(S1,ET,EQ)
      T(4)=T(4)+ET
      T(5)=T(5)+EQ
      T(2)=T(1)
      T(1)=T(1)+S1
      E(INT)=DMAX1(DABS(T(1)*RE),.1D-3)
      IF(M.LT.I1) GO TO 156
      IF(DABS(S1).GT.E(INT)) GO TO 156
      IF(S*S1.GT.0.0D0) GO TO 156
      IZ=IZ+1
      IF(IZ-I1) 158,158,188
156   IZ=0
158   S=S1
      A1=B1
      M=M+1
      GO TO 132
188   T(3)=(T(1)+T(2))/2.0D0
      E(INT)=DABS(S1)/2.0D0
      SUM(INT)=T(3)
900   CONTINUE
      SRE=SUM(1)+SUM(2)+SUM(3)+SUM(4)
      SIM=SUM(5)-SUM(6)+SUM(7)-SUM(8)
800   CONTINUE
      STOP
      END
```

```
 SUBROUTINE GAUSS1(S.TE.TEQ)
 IMPLICIT REAL*8(A-H,O-Z)
 COMMON A.B.A2,B2.R,INT
 DIMENSION C(6,6),D(6,6),V(6),H(150),U(150),Q(150),W(6,6),Z(6.6)
 DIMENSION SS(150),JJ(150),SV(150)
 DATA C(2,1),C(2,2)/2*.5D0/
 DATA C(3,1),C(3,3),C(3.2)
>/2*.27777777777777780D0,  .44444444444444440D0/
 DATA C(4,1),C(4,4),C(4,2),C(4,3)
>/2*.17392742256372700,       2*.32607257743127300/
 DATA C(5,1),C(5,5),C(5,2),C(5,4),C(5,3)
>/2*.11846344252809520D0,  2*.23931433524968300,
> .23444444444444440D0/
 DATA C(6,1),C(6,6),C(6,2),C(6,5),C(6,3),C(6,4)
>/2*.85662246189585D-1, 2*.18038078652407000, 2*.23395696728634500/
 DATA D(2,1),D(2,2)
>/.21132486540518710D0,  .78867513459481290D0/
 DATA D(3,1),D(3,2),D(3,3)
>/.11270166537925800,  .5D0,  .88729833462074200/
 DATA D(4,1),D(4,2),D(4,3),D(4,4)
>/.69431844202973D-1,  .33000947320757200,  .66999052179242800,
> .93056815579702700/
 DATA D(5,1),D(5,2),D(5,3),D(5,4),D(5,5)
>/.46910077030668D-1,  .23076534494715900,  .5D0,
> .76923465505284100,  .95308992296933200/
 DATA D(6,1),D(6,2),D(6,3),D(6,4),D(6,5),D(6,6)
>/.33765242898424D-1,  .16939530676686700,  .38069040695840100,
> .61930959304159900, .83060469323313300, .96623475710157600/
 DATA W(2.1),W(2,2)
>/1.30429030972509200,  .69570969027490800/
 DATA W(3,1),W(3,2),W(3,3)
>/.93582786914533200,  .72152314609627800,  .34264898475834000/
 DATA W(4,1),W(4,2),W(4,3),W(4,4)
>/.72536756675672400,  .62741329175577400,  .44476206890674800,
> .20245707258075200/
 DATA W(5,1),W(5,2),W(5,3),W(5,4),W(5,5)
>/.59104844942995060D0,  .53853343861999200,  .43817272503196400,
> .29390269830116200,  .13334268361737600/
 DATA W(6,1),W(6,2),W(6,3),W(6,4),W(6,5),W(6,6)
>/.49829409162680600.  .46698507307671000,  .40633485344613200,
> .32015665708669200,  .21387865199063500,  .94350672773024D-1/
 DATA Z(2,1),Z(2,2)
>/.11558710999704770D0,  .74155574714580990D0/
 DATA Z(3,1),Z(3,2),Z(3,3)
>/.56939115967007 4D-1,  .43719785275109460D0,  .86908437843247150D0/
 DATA Z(4,1),Z(4,2),Z(4,3),Z(4,4)
>/.33648268067506920-1,  .27618431387246440D0,  .63467747623463710D0,
> .92215660849205830D0/
 DATA Z(5,1),Z(5,2),Z(5,3),Z(5,4),Z(5,5)
>/.22163568807217570-1,  .18783156765244530D0,
> .46159736149626620D0,  .74833462833728130D0,  .94849392628836910D0/
 DATA Z(6,1),Z(6,2),Z(6,3),Z(6,4),Z(6,5),Z(6,6)
>/.15683406607400470-1,  .13530001165524810D0,  .38069040695840100,
> .61930959304159900,  .81742801326687520D0,  .96346129637091280D0/
 DATA RE/.5D-3/,V(1)/0.0D00/
```

```
      N=1
      M=1
      JJ(1)=2
      SV(1)=0.0D0
      Q(N)=1.0D63
      E=1.0D62
      H(N)=B-A
      U(N)=A
222   S=0.0D0
      TE=0.0D0
      TEQ=0.0D0
200   IF(Q(N).LE.E) GO TO 100
      NJ=JJ(N)
      IF(NJ-6) 30,30,500
30    DO 10 J=NJ,6
      V(J)=0.0D0
      EQ1=0.0D0
      DO 20 K=1,J
      X=U(N)+D(J,K)*H(N)
      GO TO(1,1,2,2,1,1,2,2),INT
1     IF(U(N).EQ.1.0D0) X=U(N)+Z(J,K)*H(N)
2     CALL GAUSS2(X,Y,C,D,EQ)
      EQ1=EQ1+EQ
      GO TO(3,3,4,4,3,3,4,4),INT
3     IF(U(N).EQ.1.0D0) GO TO 400
4     CALL FT(X,Y,FX)
      V(J)=V(J)+C(J,K)*H(N)*FX
      GO TO 20
400   CALL FT(X,Y,GX)
      V(J)=V(J)+W(J,K)*DSQRT(Z(J,K))*H(N)*GX
20    CONTINUE
      SS(N)=V(J)
      JJ(N)=J+1
      Q(N)=DABS(SS(N)-SV(N))
      IF(Q(N).LE.E) GO TO 100
      SV(N)=SS(N)
10    CONTINUE
500   H(N)=H(N)/2.0D0
      M=M+1
      H(M)=H(N)
      U(M)=U(N)+H(N)
      Q(M)=Q(N)
      JJ(N)=2
      JJ(M)=2
      SV(N)=0.0D0
      SV(M)=0.0D0
      GO TO 200
100   S=S+SS(N)
      SV(N)=SS(N)
      TE=TE+Q(N)
      TEQ=TEQ+EQ1
      N=N+1
      IF(N.LT.M+1) GO TO 200
      IF(TE.LE.RE*DABS(S)) GO TO 300
      N=1
      E=(TE-RE*DABS(S))/M
      GO TO 222
300   RETURN
      END
      SUBROUTINE GAUSS2(Y,S,C,D,TE)
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON A1,B1,A,B,R,INT
      DIMENSION C(6,6),D(6,6),V(6),H(150),U(150),Q(150)
      DIMENSION SS(150),JJ(150),SV(150)
      DATA RE/.5D-4/,V(1)/0.0D00/
      N=1
      JJ(1)=2
      SV(1)=0.0D0
      E=1.0D62
      Q(N)=1.0D63
      M=1
      H(N)=B-A
      U(N)=A
222   S=0.0D0
      TE=0.0D0
200   IF(Q(N).LE.E) GO TO 100
      NJ=JJ(N)
      IF(NJ-6) 30,30,500
30    DO 10 J=NJ,6
      V(J)=0.0D0
      DO 20 K=1,J
      X=U(N)+D(J,K)*H(N)
      CALL FO(X,Y,FX)
20    V(J)=V(J)+C(J,K)*H(N)*FX
      SS(N)=V(J)
      JJ(N)=J+1
      Q(N)=DABS(SS(N)-SV(N))
      IF(Q(N).LE.E) GO TO 100
      SV(N)=SS(N)
10    CONTINUE
500   H(N)=H(N)/2.0D0
      M=M+1
      H(M)=H(N)
      U(M)=U(N)+H(N)
      Q(M)=Q(N)
      JJ(N)=2
      JJ(M)=2
      SV(N)=0.0D0
      SV(M)=0.0D0
      GO TO 200
100   S=S+SS(N)
      SV(N)=SS(N)
      TE=TE+Q(N)
      N=N+1
      IF (N.LT.M+1) GO TO 200
      IF(TE.LE.RE*DABS(S)) GO TO 300
      N=1
      E=(TE-RE*DABS(S))/M
      GO TO 222
300   RETURN
      END
      SUBROUTINE PVAL(P1,P2,PI)
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON A1,B1,A2,B2,R,INT
      GO TO(1,2,1,2,3,4,3,4),INT
1     P1=PI/(2.0D0*R)
      P2=PI/(2.0D0*R)
      RETURN
2     P1=PI/(4.0D0*R)
      P2=PI/(4.0D0*R)
      RETURN
3     P1=PI/(4.0D0*R)
      P2=PI/(2.0D0*R)
      RETURN
4     P1=PI/(2.0D0*R)
      P2=PI/(4.0D0*R)
      RETURN
      END
```

```
      SUBROUTINE LVAL(P1,L)
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON A1,B1,A2,B2,R,INT
      GO TO(1,2,3,3,2,1,3,3),INT
1     DO 10 I=1,100
      L=I
      IF(L*P1.GT.1.0D0)RETURN
10    CONTINUE
      RETURN
2     DO 20 I=1,100
      L=I
      IF((2*L-1)*P1.GT.1.0D0) RETURN
20    CONTINUE
      RETURN
3     L=1
      RETURN
      END
      SUBROUTINE A1V
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON A1,B1,A2,B2,R,INT
      GO TO(2,2,1,1,2,2,1,1),INT
1     A1=0.0D0
      RETURN
2     A1=1.0D0
      RETURN
      END
```

```
      SUBROUTINE B1V(M,P1)
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON A1,B1,A2,B2,R,INT
      GO TO(1,2,1,2,2,1,2,1),INT
1     B1=DSQRT(M*P1)
      RETURN
2     B1=DSQRT((2*M-1)*P1)
      RETURN
      END
```

27 March 1981
GHP:SJG:lhz

```
      SUBROUTINE FT(X,Y,FX)
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON A1,B1,A2,B2,R,INT
      GO TO(1,2,3,4,2,1,4,3),INT
1     FX=Y*DSIN(2.0D0*R*X*X)/DSQRT(1.0D0-1.0D0/X)
      RETURN
2     FX=Y*DCOS(2.0D0*R*X*X)/DSQRT(1.0D0-1.0D0/X)
      RETURN
    3 FX=Y*DSIN(2.0D0*R*X*X)/DSQRT(1.0D0+1.0D0/X)
      RETURN
4     FX=Y*DCOS(2.0D0*R*X*X)/DSQRT(1.0D0+1.0D0/X)
      RETURN
      END
```

```
      SUBROUTINE FO(X,Y,FX)
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON A1,B1,A2,B2,R,INT
      GO TO(1,2,3,4,1,2,3,4),INT
1     FX=DSIN(R*X*X*2.0D0)*DSQRT(1.0D0/X-1.0D0)/(1.0D0-Y/X)
      RETURN
2     FX=DCOS(R*X*X*2.0D0)*DSQRT(1.0D0/X-1.0D0)/(1.0D0-Y/X)
      RETURN
3     FX=DSIN(R*X*X*2.0D0)*DSQRT(1.0D0/X-1.0D0)/(1.0D0+Y/X)
      RETURN
4     FX=DCOS(R*X*X*2.0D0)*DSQRT(1.0D0/X-1.0D0)/(1.0D0+Y/X)
      RETURN
      END
```

DISTRIBUTION LIST FOR UNCLASSIFIED TECHNICAL MEMORANDUM FILE NO. 81-82, by G. H. Peebles and S. J. Giner, dated 27 March 1981

Commander
David W. Taylor Naval Ship
Research & Development Ctr.
Department of the Navy
Bethesda, MD 20084
Attn: M. Tod Hinkel
Code 1504
(Copies 1 through 6)

Defense Technical Information Ctr.
5010 Duke Street
Cameron Station
Alexandria, VA 22314
(Copies 7 through 12)

Commander
Naval Sea Systems Command
Department of the Navy
Washington, DC 20362
Attn: T. E. Peirce
Code NSEA 63R31
(Copy No. 13)

Director
Applied Research Laboratory
The Pennsylvania State University
Post Office Box 30
State College, PA 16804
Attn: ARL/PSU Library
(Copy No. 14)

Director
Applied Research Laboratory
The Pennsylvania State University
Post Office Box 30
State College, PA 16804
Attn: S. J. Giner
(Copy No. 15)

Director
Applied Research Laboratory
The Pennsylvania State University
Post Office Box 30
State College, PA 16804
Attn: GTWT Files
(Copy No. 16)

Director
Applied Research Laboratory
The Pennsylvania State University
Post Office Box 30
State College, PA 16804
Attn: L. R. Hettche
(Copy No. 17)

Director
Applied Research Laboratory
The Pennsylvania State University
Post Office Box 30
State College, PA 16804
Attn: B. R. Parkin
(Copy No. 18)

Director
Applied Research Laboratory
The Pennsylvania State University
Post Office Box 30
State College, PA 16804
Attn: R. Stern
(Copy No. 19)

Mr. Glenn H. Peebles
12179 Leven Lane
Los Angeles, CA 90049
(Copy No. 20)